CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Hasanuddin University Repository

Memory Sharing Management on Virtual Private Server

Muh. Niswar
Department of Electrical Engineering
Hasanuddin University
Makassar, INDONESIA

Elly Warni
Department of Electrical Engineering
Hasanuddin University
Makassar, INDONESIA

A. Aulia Sabri
Department of Electrical Engineering
Hasanuddin University
Makassar, INDONESIA

Muh. Nur Musa
Department of Electrical Engineering
Hasanuddin University
Makassar, INDONESIA

***Abstract*—Data center required to guarantee high availability of service to users. Recently, most data center use virtualization technologies to provides high availibility service. The virtualization technology allow a physical server to be virtualized into multiple virtual servers. This virtual server is logically working separately although physically located on the same hardware. When several virtual servers are created and they have heavy workload, server with limited resources must able to manage the use of resource among vitual server. We propose a memory resource sharing among virtual machine in order to achieve high availibility services. A virtual server that has an overload can get memory resources from another virtual server that has less workload and it has memory unused. In addition, the virtual server can also use the Hardware Node RAM unused in case that all virtual servers have heavy workload and exeperince memory exhaust. Thus, high availibility services can be achieved.**

Keywords—virtual server; memory; resource sharing;

I. INTRODUCTION

Today, most of organizations/industries rely on a computer network to enhance their productivity. Server as one of the main components in the computer network plays an important role in providing services to users or customers. The server must be able to meet the demands of the users in order to support users work needs. Therefore, high availibility (HA) becomes a requirement to ensure the availability of services to users.

Recently, most data center use virtualization technologies to provides high availibility service. The virtualization technology allow a physical server to be virtualized into multiple virtual private servers (VPS). This VPS is logically working separately although physically located on the same hardware. The need of a reliable server is not followed by efficient utilization of servers' resources. Sometimes resources are not fully utilized when the allocation of resources is fixed. Consequently, when a VPS requires more resources, the allocation of unused resources from other VPS cannot be used. Therefore, we need a method to dynamically allocate resources among VPSs.

In this research, we adopt VMware DRS (distributed resource scheduling) concept. VMware uses VMware DRS to create high availability service on server cluster. VMware DRS uses VMotion to provide automated resource optimization and virtual machine placement and migration. DRS provides automatic initial virtual machine placement on any of the hosts in the cluster, and also makes automatic resource relocation and optimization decisions as hosts or virtual machines are added or removed from the cluster. [1]

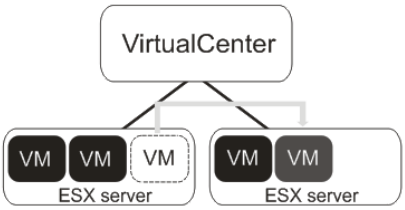

Fig. 1. DRS Scheduler of Clusters in VirtualCenter.

We simplify this concept to memory sharing management. In VMware DRS resource is managed by moving virtual machine from one cluster to another cluster using VMotion. In our memory sharing management, resource (memory) is managed by lending memory allocation from one virtual machine to another virtual machine (section II). So, virtual machine that need additional memory can get it. Therefore, all process can served by VPS.

We use OpenVZ [2] as our virtualization environment. The OpenVZ virtualization provides near base Linux performance (low overhead) [3], which is a great solution to run multiple VPS on one physical server. This study focuses on memory resources sharing among VPS that created on top of single physical server using OpenVZ virtualization environment.

II. PROPOSED MEMORY SHARING METHODS

In this section, we give overview of our proposed method for memory sharing management. Our proposed method implemented in bash scripting. Our method monitors every running VPS memory usage, detect the VPS that requires more memory allocation, and lease memory resources from one VPS to needing VPS. The following is functions that used in our method.

A. *get_resource function*

get_resource function is used to obtain RAM states of running VPSs. These states are then used to analyze the need of resource sharing between VPS. These states consist of the amount of RAM in use, the allocation of RAM, swap unused, and the allocation of swap.

B. *Is_full function*

Is_full function is used to specify whether a particular VPS need more RAM allocation or not. Thus, the system is able to decide whether to allocate RAM to a VPS or not. Operating systems such as BSD Unix did reclaiming when RAM free is below 5% and will continue to reclaiming until the RAM free up to 7% [4]. When memory resource exhaustion happens in Linux, Out-Of-Memory (OOM) killer selects a process to eliminate to free up memory resources. This feature is used to protect the Linux kernel from becoming unresponsive [5]. Thus, to create high availability service, we should avoid memory exhaustion to happen by increasing memory allocation for VPS with high memory usage. On the VMwareESX server there are four thresholds to perform reclaiming they are high, soft, hard, and low with the percentages of each 6%, 4%, 2%, and 1% of RAM free. Therefore, our method elected 10% RAM free as a threshold to declare a VPS is on a busy state. The pseudo code of our method is as follows:

```
limit = 90% mem_limit
IF limit > mem_used THEN
        state = false // no need for more memory allocation
ELSE
        status = true // need more memory allocation
ENDIF
```

C. *To_share function*

to_share function is used to determine whether a VPS can lend RAM to needing VPS or not. It prevents a VPS to become lack of free RAM after lending its RAM to other VPS. The algorithm is as follows:

```
treshold_to_share = amount_to_be_shared + 20% used_ram
IF free_mem> treshold_to_share THEN
        share_state = true // the VPS may lend its RAM allocation
ELSE
        share_state = false // the VPS may not lend its RAM allocation
ENDIF
```

D. *To_return function*

to_return function is used to determine whether the VPS which borrowed RAM from other VPS may return the RAM to VPS which lend it RAM. It prevents VPS be busy again shortly after returning borrowed RAM. The pseudocode of our method-is as follows:

```
treshold_to_return = 80% (allocated_ram - borrowed_resource)
IF used_ram <= treshold_to_return THEN
        status_return = true // May return the memory
ELSE
share_status = false // May not return the memory
ENDIF
```

III. EXPERIMENTS AND RESULTS

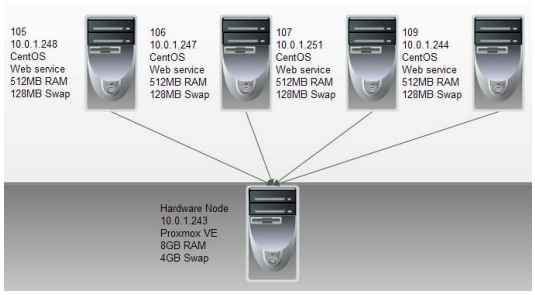

Fig. 2. Topology used in experiments.

A. *Experimental*

We conduct experiments to investigate the effectiveness of our proposed memory sharing management. We implement our method on a physical server containing four VPSs (VPS105, VPS106, VPS107, and VPS109) running web service. Our method run as background process (daemon) named *vzrsmd* to allocate RAM dynamically when VPS needed. Each VPS allocated 512 MB of RAM and 128 MB of swap. We use Apache JMeter to generate VPSs and test reporting.

Tests carried out in two major phases, without running the program and with running the program. The first phase aims to find the threshold of VPS capabilities and its response when the number of requests exceeds the capacity. The second phase is carried out to test the performance of the server in performing the sharing RAM among VPSs or between the VPS and the hardware node. During the test, the use of RAM, swap, and RAM allocation changes will be logged by a script named resmon.sh.

B. *Results and Analysis*

From experiments we obtain that server reach its RAM usage limit on 150 user threads, ramp-up time 20, and repeat 5 times.

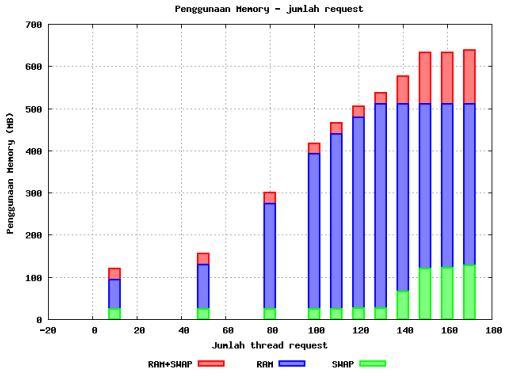

Fig. 3. Memory usage – number of thread requests

From figure 3, we know that VPS reach maximum RAM usage when handling 150 threads requests. The memory usage details when VPS handling 150 threads requests as follow:

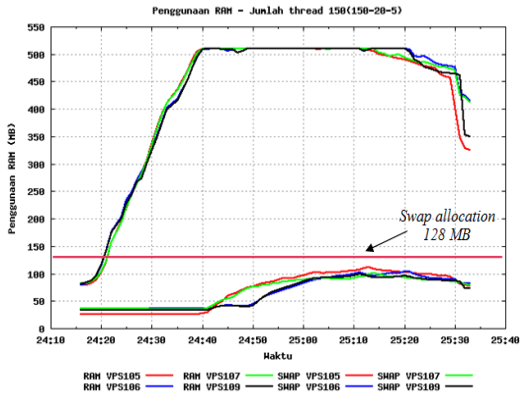

Fig. 4. VPS Memory usage – time

The second experiment performed to determine the VPS response to requests that exceed the threshold. Therefore, the test was conducted with 170 user threads, ramp-up time 20, and repeat 5 times. Figure 5 shows the result of the second experiment.

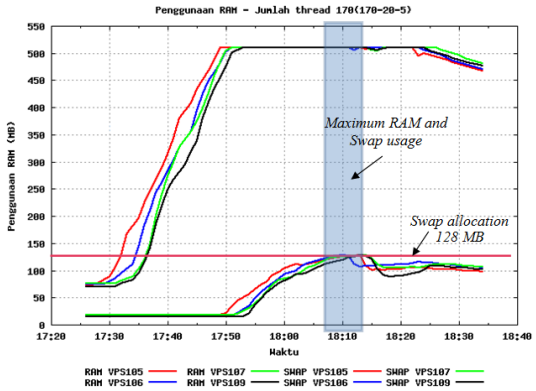

Fig. 5. RAM and swap usage handling 170 threads

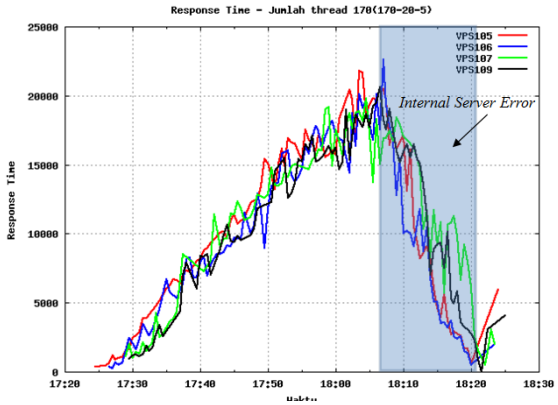

Fig. 6. VPS response time

From the figure, we can see that VPSs were experiencing out of memory situation while handling 170 threads requests.

Figure 7 shows the error page received by user showing internal server error experienced on VPS.

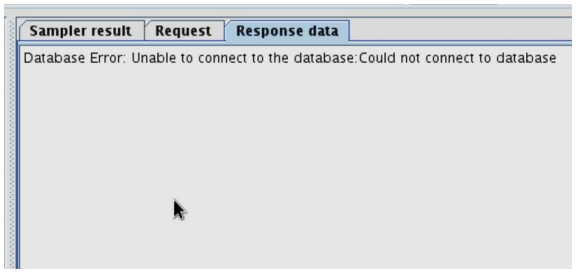

Fig. 7. Error page received

From our investigation, VPS killed the MySQL process because it experiencing out of memory situation. After knowing the VPS performance limit, we run our method to see performance difference on VPS and to investigate our method ability to dynamically allocate RAM to corresponding VPS.

We conduct an experiment by sending http request to VPS 106 and VPS 109 and leaving VPS 105 and VPS 107 to be idle. We use 170 threads requests. Here is the result:

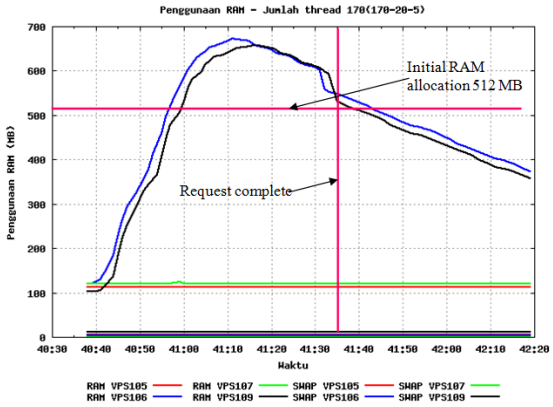

Fig. 8. VPS memory usage – time

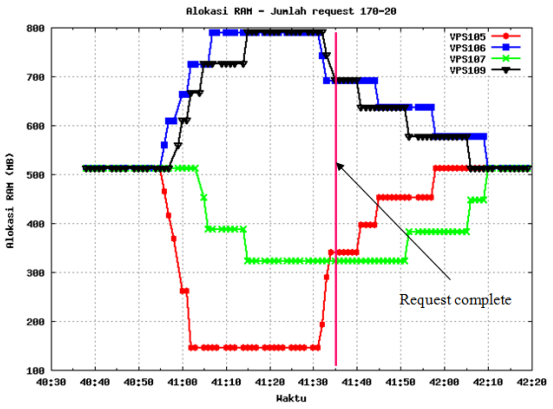

Fig. 9. RAM allocation on each VPS

From figure 8, we can see that both VPS 106 and 109 use RAM above the limit we set before. This is because our method allocate more RAM to VPS 106 and VPS 109 because it detect that both VPSs need more RAM allocation. When the

VPS RAM usage reaches the limit (being busy), then our method will find a VPS that can be borrowed its RAM based on the smallest ID (VPS 105) from the list of active VPS. Figure 9 shows the VPS RAM allocation when the experiment is conducted.

After the VPS 105 is no longer eligible *to_share* function then VPS 107 RAM is used. This happens because the VPS 106 is declared as busy to lend its RAM allocation. VPS performance also increased as evidenced by the average of VPS response time of 6:49 seconds. At the end of the graph shown all VPS back to the initial allocation of RAM.

The last experiment, we stress all VPSs while running the program. This experiment is to see how our method allocates RAM to VPS while there are no VPS able to lend their RAM (because they all need that). We use 170 requests to see the difference with the same test without running the program. Figure 10, 11, Figure 10, 11, 12 show the result of the last experiment.

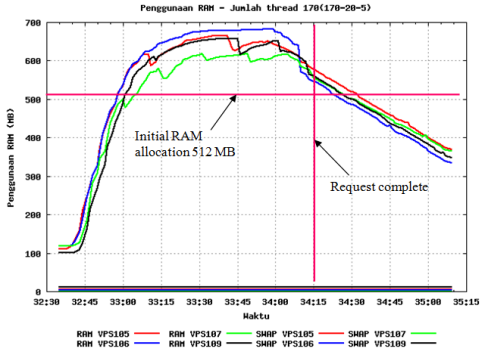

Fig. 10. RAM usage on each VPS

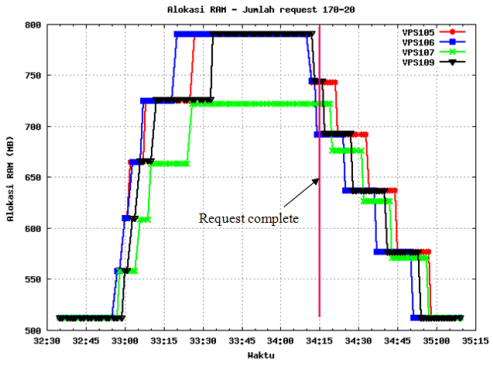

Fig. 11. RAM allocation on each VPS

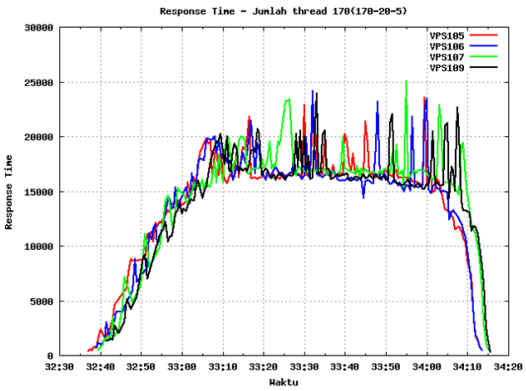

Fig. 12. Response time on each VPS

On this last experiment, no VPS experiencing out of memory situation. It is different from experiment without running our program before. The mySQL process did not get killed so all pages returned not experiencing error.

IV. CONCLUSION

This paper presents a method to share memory allocation among VPSs. We proposed a method that allows VPS to lend their memory resources to another VPS experiencing memory exhaustion. Our experimental results show that our method can increase VPS performance and improve the the RAM utilization. In case that all VPSs are busy and experiencing RAM exhaustion, then VPS get the RAM from hardware node.

REFERENCES

[1] Resource Management with VMware DRS, ver. 1.1, VMware Inc., Palo Alto, CA, 2006, pp. 5-6.

[2] "OpenVZ Linux Containers Wiki." Internet: http://wiki.openvz.org/Main_Page, May 7, 2013 [Dec. 22, 2012]

[3] Pradeep Padala, Xiaoyun Zhu, Zhikui Wang, Sharad Singhal, and Kang G. Shin, "Performance Evaluation of Virtualization Technologies for Server Consolidation," HP Laboratories, page 13, April 2007.

[4] Carl A. Waldspurger. "Proceedings of the 5th Symposium on Operating Systems Design and Implementation : Memory Resource Management in VMware ESX Server". Boston, USENIX Association, page 14, December 2002.

[5] Ron Brightwell, "On the Appropriateness of Commodity Operating Systems for Large-Scale, Balanced Computing Systems," in *International Parallel and Distributed Processing Symposium*, Nice, 2003, page 7.